**ABSTRACT**

**iTask – To-Do List Manager**

Task management is a critical aspect of productivity, yet many users struggle with **inefficient tracking, missed deadlines, and lack of structured data storage**. Existing solutions often suffer from **poor database design,** leading to **slow retrieval times, data inconsistency, and limited scalability**. Additionally, most task managers **lack integration with external scheduling tools** like **Google Calendar**, making it difficult for users to receive timely reminders.

To address these challenges, we propose a **To-Do List Task Manager** that leverages **React.js for the frontend, Node.js for the backend, and MySQL for structured data storage**. Our system efficiently organizes tasks, assigns deadlines, and provides real-time reminders by integrating with the **Google Calendar API**. The database is designed using an **Entity-Relationship (ER) model,** ensuring **data normalization, referential integrity, and optimized indexing for faster query performance.**

The application implements **authentication** to ensure **secure user access** and supports **role-based access control** for team collaboration. Tasks are stored in a **normalized MySQL database**, allowing **quick retrieval and minimal redundancy**. Additionally, **cron jobs** are used to automate reminder notifications, reducing the risk of missed deadlines.

By combining **a well-structured database, efficient backend logic, and real-time external integrations**, this project offers a **scalable, high-performance, and user-friendly task management solution** that significantly improves productivity and organization.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Title** | **Page no.** |
|  | ABSTRACT | 1 |
| 1 | Problem understanding, Identification of Entity and Relationships, Construction of DB using ER Model for their project | 5-10 |
| 2 | Design of Relational Schemas, Creation of Database and Tables for their project |  |
| 3 | Writing complex queries based on the concepts of constraints, sets, joins, views, Triggers, and Cursors |  |
| 4 | Analyzing the pitfalls, identifying the dependencies, and applying normalizations |  |
| 5 | Implementation of concurrency control and recovery mechanisms. |  |

**TABLE OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Name** | **Page no.** |
| 1 | Figure 1.1: ER Diagram of iTask Website | 10 |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |

**CHAPTER 1**

**PROBLEM UNDERSTANDING, IDENTIFICATION OF ENTITY AND RELATIONSHIPS, CONSTRUCTION OF DB USING ER MODEL FOR iTask: A TO-DO LIST PLATFORM**

* 1. **Introduction**
     1. **Motivation**
     2. **Scope**
  2. **Problem Statement**
  3. **Identification of Entity and Relationships**
  4. **Construction of DB Using ER Model for iTask Website**

**1.1 Introduction**

**1.1.1 Motivation**

Task management is crucial for productivity, but users often struggle to keep track of their responsibilities, leading to missed deadlines and inefficiencies. Many existing solutions lack a well-structured database or seamless integration with external tools like Google Calendar. Our motivation is to design a **database-driven task manager** that enables efficient task tracking, structured data storage, and timely reminders.

**1.1.2 Scope**

The scope of this project encompasses the design and implementation of a **database management system (DBMS)** for the iTask a to-do list platform. This includes:

* Optimized Relational Database: Uses MySQL with normalized tables for fast data retrieval and integrity.
* Structured Task Organization: Defines clear entity relationships between Users, Tasks, Teams, and Reminders.
* Automated Reminder System: Uses cron jobs for scheduled notifications and deadline alerts.
* Scalability & Performance: Optimized SQL queries, indexing, and caching for high-speed performance.
* Future Expansion: Supports potential AI task prioritization, mobile app integration, and analytics.

**1.2 Problem Statement**

Managing tasks effectively is a challenge due to **inefficient tracking and missed deadlines.** Existing solutions often lack structured databases, leading to **data inconsistency** and **performance issues**. Moreover, many task managers do not integrate with external tools like Google Calendar, preventing users from receiving timely reminders.

### **Project Objectives:**

* **Efficient Data Management:** Optimize task storage using a structured relational database.
* **Task and Reminder System:** Enable users to schedule tasks with automated notifications.
* **User Authentication and Role Management:** Secure user data and provide controlled access.
* **Google Calendar Integration:** Sync tasks with Google Calendar for real-time reminders.
* **Scalability and Performance Optimization:** Design the system for future growth.

**1.3 Identification of Entities and Relationships**

The ER model represents a **Task Management System** where users, teams, tasks, comments, reminders, and attachments are interrelated. The entities and their attributes are as follows:

### **Entities and Their Attributes**

1. Users

* user\_id (Primary Key)
* name
* email
* phone\_no1
* phone\_no2

1. Teams

* team\_id (Primary Key)
* name
* created\_by (Foreign Key referencing Users)
* created\_at

1. User\_Teams

* user\_id (Primary Key, Foreign Key referencing Users)
* team\_id (Primary Key, Foreign Key referencing Teams)

1. Tasks

* task\_id (Primary Key)
* title
* status
* created\_by (Foreign Key referencing Users)
* team\_id (Foreign Key referencing Teams)

1. Task\_Comments

* comment\_id (Primary Key)
* task\_id (Foreign Key referencing Tasks)
* created\_by (Foreign Key referencing Users)
* content

1. Task\_Attachments

* attachment\_id (Primary Key)
* task\_id (Foreign Key referencing Tasks)
* file\_name
* file\_path

1. Reminders

* reminder\_id (Primary Key)
* task\_id (Foreign Key referencing Tasks)
* user\_id (Foreign Key referencing Users)
* message
* is\_read

**Relationships**

The relationships between the entities are as follows:

1. Users and Teams
   * A user can create a team.
   * A team belongs to one user (created\_by).
   * A team consists of multiple users (Members).
2. Users and Tasks
   * A user creates tasks.
   * A task belongs to one user (created\_by).
   * A task belongs to one team.
3. Users and Task Comments
   * A user writes comments on tasks.
   * A comment belongs to a specific task.
4. Users and Task Attachments
   * A user uploads attachments to a task.
   * A task can include multiple attachments.
5. Tasks and Reminders
   * A task generates multiple reminders.
   * A reminder is received by a specific user.
6. Teams and Tasks
   * A task belongs to one team.
   * A team manages multiple tasks.

**1.4 Construction of Database Using ER Model for iTask**

### **Data Flow in the System:**

1. User logs in and creates a task.
2. Task details are stored in the **Tasks Table**.
3. If enabled, the task syncs with **Google Calendar**, creating an event.
4. The system schedules **reminders** from the **Reminders Table**.
5. The user receives a notification before the task deadline.
6. The user marks the task as completed; the status updates in the **Tasks Table**.
7. If Google Calendar is enabled, the task status updates there as well.

### **Entity-Relationship (ER) Model:**

The ER model will include:

* **Users (Primary Key: user\_id)**
* **Tasks (Primary Key: task\_id, Foreign Key: user\_id, team\_id)**
* **Teams (Primary Key: team\_id)**
* **Reminders (Primary Key: reminder\_id, Foreign Key: task\_id)**
* **Google Calendar Events (Primary Key: event\_id, Foreign Key: task\_id)**
* **Attachments, Comments, and Task History Tables** with relationships to **Tasks**

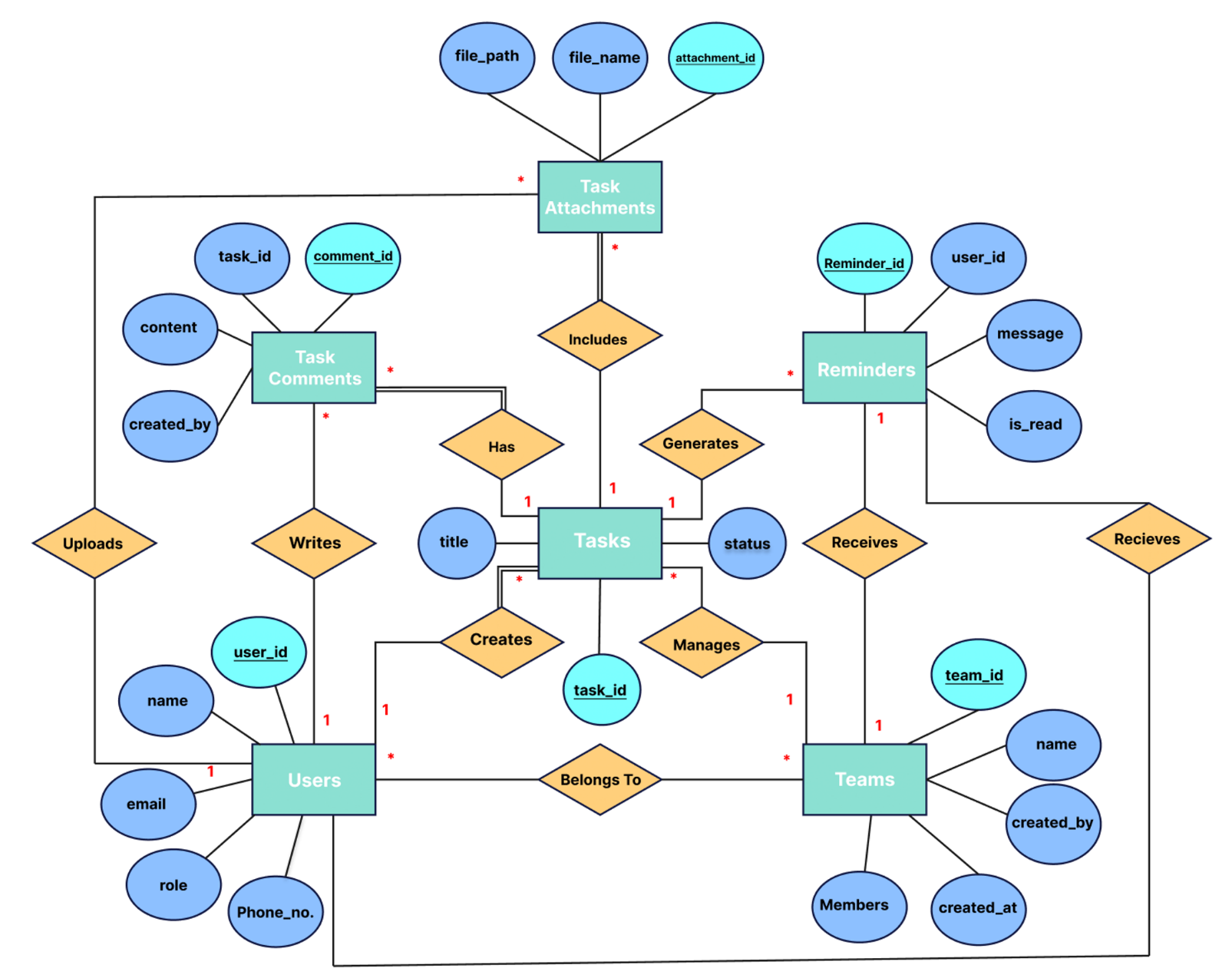


Figure 1.1: ER diagram of iTask – To-Do list manager